

# Clase 2 — 24.10.25

#VSC

Desarrollo de interfaces. JAVASCRIPT, JQUERY, REALIDAD VIRTUAL

Clase 2 — 24/10/2025

Tema: Primeros pasos con Javascript | Accesibilidad y Usabilidad

Profesora: Sara Gonzalo

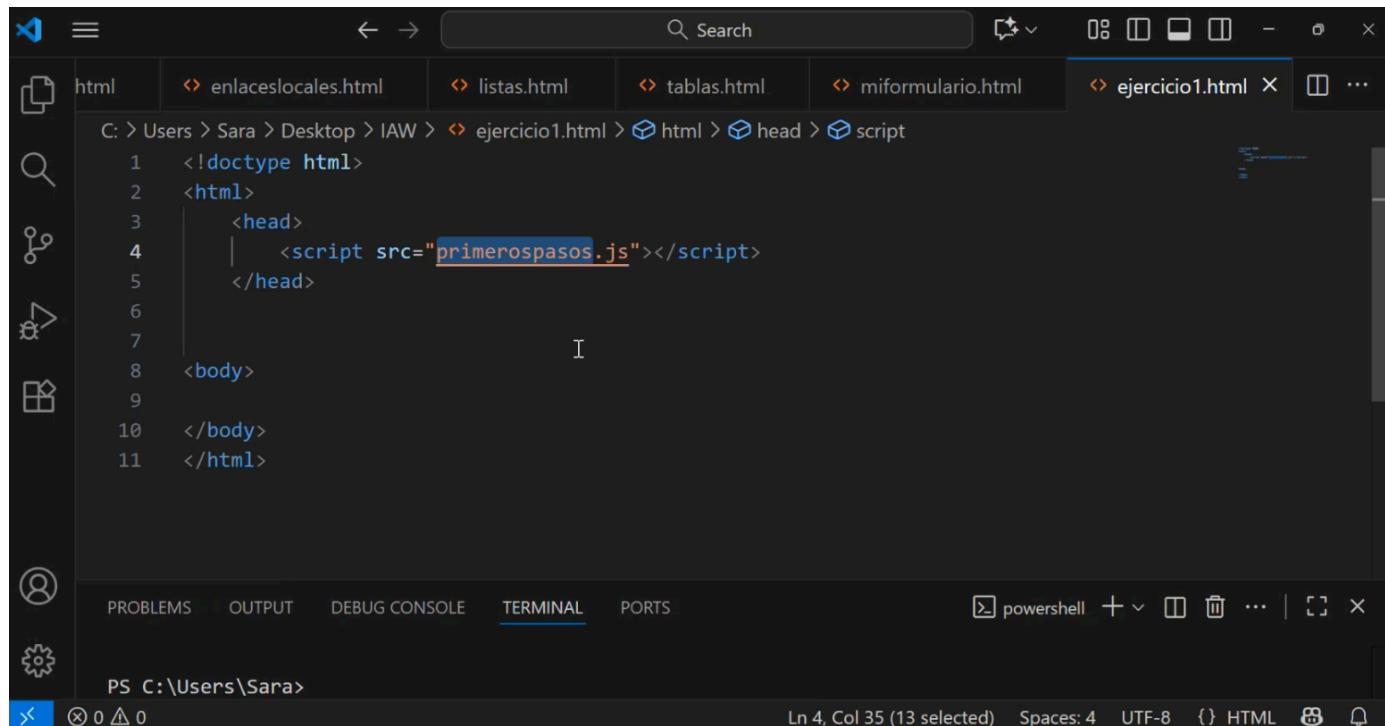
## Desarrollo de Interfaces (Clase erronea ASIR)

### Entorno de trabajo: Visual Studio Code (VSC)

Para el desarrollo de interfaces trabajaremos en **Visual Studio Code**, que será nuestro **IDE principal** (entorno de desarrollo integrado).

El trabajo con **JavaScript (JS)** se realizará en **archivos externos**, que se vinculan desde el documento HTML mediante la etiqueta:

```
<script src="primerospasos.js"></script>
```



The screenshot shows the Visual Studio Code interface. On the left is the sidebar with file icons and a list of files: html, ejercicioslocales.html, listas.html, tablas.html, miformulario.html, and ejercicio1.html. The main editor area displays the following HTML code:

```
C: > Users > Sara > Desktop > IAW > ejercicio1.html > html > head > script
1  <!doctype html>
2  <html>
3    <head>
4      <script src="primerospasos.js"></script>
5    </head>
6
7    <body>
8
9
10   </body>
11   </html>
```

Below the editor are several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. At the bottom, there's a terminal window showing a PowerShell prompt: PS C:\Users\Sara>. The status bar at the bottom right indicates: Ln 4, Col 35 (13 selected) Spaces: 4 UTF-8 {} HTML.

Esto permite separar la lógica del programa del contenido HTML, mejorando la organización y mantenimiento del código.

### Comentarios en JavaScript

Existen dos tipos de comentarios:

```
// Comentario en una sola línea
```

```
/* Comentario  
en bloque o multilínea */
```

Los comentarios **no se ejecutan** y sirven para documentar el código o desactivar temporalmente partes del mismo.

## ◆ Variables y tipos de datos

JavaScript **no es un lenguaje tipado** (a diferencia de Java o C#).

Esto significa que una misma variable puede almacenar distintos tipos de datos a lo largo del programa.

### Formas de declarar variables:

```
let nombre;    // variable local  
var edad;     // variable global
```

Cláusula	Alcance	Características
let	Bloque/local	Recomendado. Optimiza recursos del sistema y evita conflictos.
var	Global	Afecta a todo el script. Puede provocar errores si se reutiliza.

### Tipos de datos primitivos

Tipo	Ejemplo	Descripción
String	"Hola"	Texto entre comillas.
Number	25 , 2.34	Números enteros o decimales.
Boolean	true , false	Valores lógicos.
Undefined	variable sin valor asignado.	let x;
Null	ausencia intencionada de valor.	let y = null;
Object	{nombre: "Eva", edad:25}	Estructura con propiedades.
Array	[1,2,3,4]	Lista ordenada de valores.

## ◆ Reglas para nombrar variables

1. No usar **caracteres especiales**: @ # ; . ñ
2. No empezar con números.
3. Evitar **palabras reservadas** del lenguaje.
4. Utilizar nombres descriptivos: nombreUsuario , totalVenta , etc.

## ◆ Mostrar información desde JavaScript

Para escribir directamente en la página web se usa el método `document.write()`:

```
document.write("Texto desde JavaScript");  
document.write("El valor de numero1 es: " + numero1);  
document.write("<strong>El valor de cansado es:</strong> " + cansado + "<br>");
```

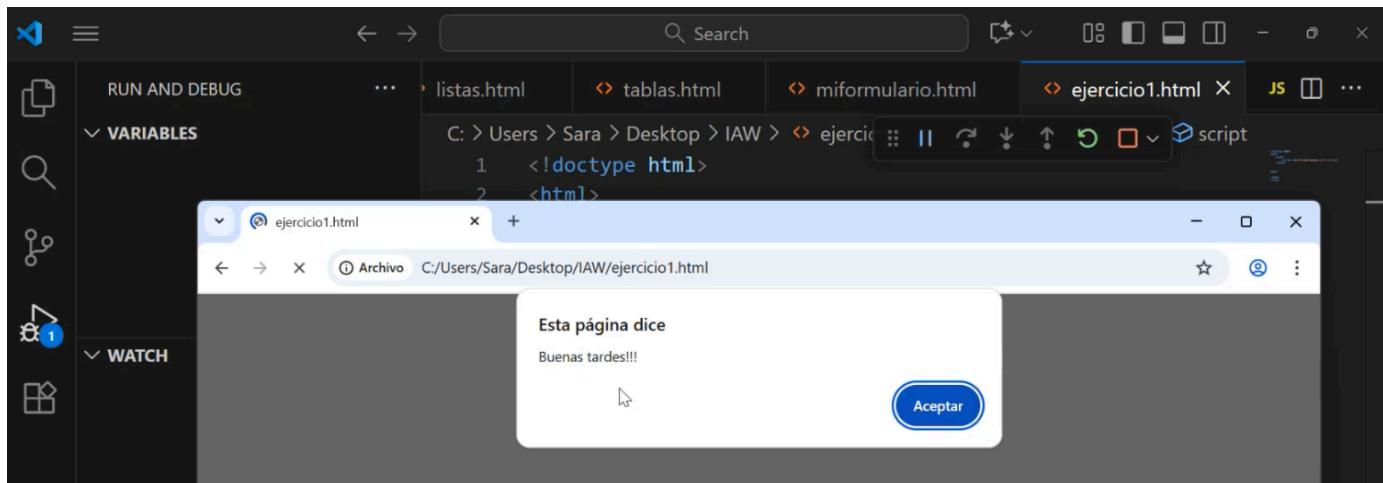
Este método permite incluir etiquetas HTML dentro del texto mostrado.

## ◆ Ventanas emergentes del objeto window

El objeto `window` hace referencia a la **ventana del navegador**.

Con él podemos mostrar o solicitar información al usuario:

Método	Descripción	Ejemplo
<code>alert()</code>	Muestra un mensaje.	<code>window.alert("Buenas tardes!!!");</code>
<code>prompt()</code>	Solicita un valor al usuario.	<code>window.prompt("Introduce un valor:");</code>
<code>confirm()</code>	Solicita confirmación (sí/no).	<code>window.confirm("¿Deseas continuar?");</code>



### Ejemplo práctico:

```
window.alert("Buenas tardes!!!");
let edad = window.prompt("Introduce tu edad:");
```

Por defecto, el método `prompt()` devuelve valores **de tipo texto** (string).

## ◆ Conversión de datos

Para convertir los valores obtenidos en números:

```
parseInt() // convierte a número entero
parseFloat() // convierte a número decimal
```

Cuando el `+` se usa entre números y cadenas, JS convierte todo a texto.

Esto genera errores comunes en principiantes.

Ejemplo:

```
let num1 = 10; let num2 = "20";
let resultado = num1 + num2;
// "1020" (concatenación, no suma)
```

Para evitarlo:

```
let resultado = num1 + parseInt(num2);
// 30
```

Añadir esta nota ayuda a entender por qué es necesario usar `parseInt()`.

## Ejemplo:

```
let num1 = parseInt(window.prompt("Introduce valor 1:"));
let num2 = parseInt(window.prompt("Introduce valor 2:"));
let decimal = parseFloat(window.prompt("Introduce valor decimal:"));
```

## ◆ Operadores aritméticos

Operador	Descripción	Ejemplo	Resultado
+	Suma	5 + 3	8
-	Resta	5 - 3	2
*	Multiplicación	5 * 3	15
/	División	10 / 2	5
%	Módulo (resto)	10 % 3	1

## Ejemplo de uso:

```
let num1 = parseInt(window.prompt("Introduce valor 1:"));
let num2 = parseInt(window.prompt("Introduce valor 2:"));

let suma = num1 + num2;
window.alert("El resultado de la suma es: " + (num1 + num2));
```

- ◆ Importante: los **paréntesis** determinan la prioridad operacional.

Si no se usan, el texto se concatenará antes de realizar la suma.

## ◆ Operadores de asignación

Operador	Significado	Ejemplo	Resultado
=	Asignación	num1 = 25	25
+=	Suma y asigna	num1 += 10	35
-=	Resta y asigna	num1 -= 5	30
*=	Multiplica y asigna	num1 *= 2	60
/=	Divide y asigna	num1 /= 10	6
%=	Módulo y asigna	num1 %= 2	0

⚠ Los textos solo pueden **sumarse (concatenarse)** con `+`, pero **no restarse, dividirse ni multiplicarse**.

## ◆ Estructuras condicionales

Las estructuras condicionales permiten **ejecutar distintas acciones según una condición**.

### Sintaxis general del `if`:

```
if (condición) {
    // Código si la condición se cumple
} else {
```

```
// Código si no se cumple
}
```

### Ejemplo: comprobar si un número es par o impar

```
let dato = parseInt(window.prompt("Introduce un número:"));

if (dato % 2 == 0) {
    window.alert("El número es par");
} else {
    window.alert("El número es impar");
}
```

## ◆ Operadores de comparación

Operador	Significado	Ejemplo	Resultado
==	Igualdad de valores	5 == "5"	true
!=	Diferente	5 != 3	true
>	Mayor que	5 > 3	true
<	Menor que	2 < 8	true
>=	Mayor o igual que	5 >= 5	true
<=	Menor o igual que	4 <= 5	true

## ◆ Ejemplo con condicionales anidados

```
let dato = parseInt(window.prompt("Introduce un valor entero:"));

if (dato == 0) {
    window.alert("El dato es igual a cero");
} else if (dato < 0) {
    window.alert("El dato es menor que cero");
} else {
    window.alert("El dato es mayor que cero");
}
```

## ◆ Operadores lógicos

Operador	Descripción	Ejemplo	Resultado
&&	AND (y) — ambas condiciones deben cumplirse	(x > 0 && y > 0)	true si ambas son verdaderas
^		^	OR (o) — basta con que una se cumpla
!	NOT (no) — invierte el resultado	!(x > 0)	true si x es menor o igual a 0

### Ejemplo:

```
let edad = parseInt(window.prompt("Introduce tu edad:"));
```

```

if (edad >= 18 && edad <= 65) {
    window.alert("Edad laboral activa");
} else {
    window.alert("Fuera del rango laboral");
}

```

## ◆ Ejemplo completo de la clase

**Archivo:** primerospasos2.js

```

C: > Users > Sara > Desktop > IAW > JS primerospasos2.js > ...
1 //Se definen las variables y asignamos valor
2
3 let num1,num2;
4 let suma,resta,multi,division,modulo;
5
6 num1=parseInt(window.prompt("Introduce valor1:"));
7 num2=parseInt(window.prompt("Introduce valor2:"));
8
9 //suma
10 suma=num1+num2;
11 window.alert("El resultado de la suma es: " + (num1+num2));
12
13 //resta
14 resta=num1-num2;
15 window.alert("El resultado de la resta es: " + resta);

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    ...    Filter (e.g. text, !exclude, \esca...    Open ejercicio2.html    🔍 | ⚙️ X

Ln 7, Col 47 (32 selected)    Spaces: 4    UTF-8    CRLF    {} JavaScript    🌐    📡

```
2
3 let num1,num2;
4 let suma,resta,multi,division,modulo;
5 let decimal;
6
7
8 num1=parseInt(window.prompt("Introduce valor1:"));
9 num2=parseInt(window.prompt("Introduce valor2:"));
10 decimal=parseFloat(window.prompt("Introduce valor decimal:")); I
11
12
13 //suma
14 suma=num1+num2;
15 window.alert("El resultado de la suma es: " + (num1+num2));
PROBLEMS OUTPUT DEBUG CONSOLE ... Filter (e.g. text, !exclude, \esca... Open ejercicio2.html x
at <anonymous> (c:\Users\Sara\Desktop\IAW\primerospasos2.js:19:58)
x Search C: > Users > Sara > Desktop > IAW > JS primerospasos2.js > ...
42 //igual a cero
43 //mayor que cero
44 //menor que cero
45
46 let dato2=parseInt(window.prompt("Introduce un valor entero"));
47
48 if(dato1==0)
49 {
50     window.alert("El dato es igual a cero");
51 }
52 else if(dato1<0)
53 {
54     window.alert("El dato es menor que cero");
55 }
56 else
57 {
58     window.alert("El dato es mayor que cero");
59 }
```

```
// Se definen las variables
let num1, num2, suma, resta, multi, division, modulo, decimal;

// Solicitamos valores
num1 = parseInt(window.prompt("Introduce valor 1:"));
num2 = parseInt(window.prompt("Introduce valor 2:"));
decimal = parseFloat(window.prompt("Introduce valor decimal:"));

// Operaciones
suma = num1 + num2;
window.alert("El resultado de la suma es: " + (num1 + num2));

// Comprobación de número positivo, negativo o cero
let dato = parseInt(window.prompt("Introduce un número entero:"));

if (dato == 0) {
    window.alert("El dato es igual a cero");
} else if (dato < 0) {
    window.alert("El dato es menor que cero");
} else {
```

```
    window.alert("El dato es mayor que cero");
}
```

## Desarrollo de Interfaces (Clase 2 DAM)

### 1 Objetivos de la unidad

- Comprender la importancia de la **usabilidad** y la **accesibilidad** en el desarrollo de interfaces.
- Analizar los **principios heurísticos de Nielsen** y las **pautas WCAG del W3C**.
- Detectar **barreras de acceso** y proponer soluciones basadas en estándares web.
- Aplicar buenas prácticas de **navegación, estructuración de contenido y verificación** de accesibilidad.

### 2 ¿Qué es la usabilidad?

La **usabilidad web** se refiere a la facilidad con la que los usuarios pueden utilizar una página para cumplir sus objetivos.

Su propósito es mejorar la **experiencia del usuario (UX)** y fomentar el **deseo de volver a usar** la aplicación o sitio.

Una interfaz puede funcionar técnicamente, pero si el usuario no sabe utilizarla, el error es igual de grave que si la app no funcionara.

#### ◆ Conceptos clave

- **Usabilidad ≠ Accesibilidad:** la primera busca facilitar el uso; la segunda, garantizar el acceso.
- **Buena usabilidad** implica:
  - Interacción sencilla e intuitiva.
  - Experiencia fluida y agradable.
  - Retroalimentación clara del sistema.

#### ◆ Beneficios

- Aumenta la satisfacción del usuario.
- Mejora la comunicación y la interacción.
- Incrementa el tráfico y las conversiones.
- Disminuye la tasa de rebote.
- Fideliza usuarios y promueve recomendaciones.

### 3 Principios de Nielsen

Los **10 principios heurísticos de Jakob Nielsen** orientan el diseño de interfaces centradas en el usuario.

Nº	Principio	Ejemplo / Aplicación práctica
1	<b>Visibilidad del estado del sistema</b>	Mostrar barras de carga, notificaciones o confirmaciones al realizar acciones.

Nº	Principio	Ejemplo / Aplicación práctica
2	<b>Relación entre el sistema y el mundo real</b>	Usar iconos y lenguaje familiar (ej. ícono de papelera = eliminar).
3	<b>Control y libertad del usuario</b>	Permitir deshacer y rehacer acciones como en editores de texto o formularios.
4	<b>Consistencia y estándares</b>	Mantener la misma disposición de botones y mensajes en toda la web.
5	<b>Prevención de errores</b>	Validar formularios antes de enviarlos para evitar fallos de usuario.
6	<b>Reconocimiento mejor que recuerdo</b>	Mostrar menús visibles sin obligar al usuario a memorizar rutas.
7	<b>Flexibilidad y eficiencia de uso</b>	Atajos de teclado o personalización de funciones para usuarios avanzados.
8	<b>Diseño estético y minimalista</b>	Evitar información innecesaria que distraiga del objetivo principal.
9	<b>Reconocer, diagnosticar y recuperarse de errores</b>	Mensajes de error claros y sin códigos ("Contraseña incorrecta").
10	<b>Ayuda y documentación</b>	Incluir sección de ayuda, soporte o guía rápida accesible.

Ejemplo práctico: en una aplicación de compras, avisar al usuario del progreso del pedido y permitir cancelar o editar el carrito cumple varios de estos principios simultáneamente.

## 4 Barreras identificadas

### ◆ 5.1 Información fácilmente accesible

La **usabilidad** es condición necesaria, pero no suficiente, para una buena accesibilidad.

- Usabilidad → se centra en el **usuario objetivo**.
- Accesibilidad → busca incluir al **máximo número posible de usuarios**, incluyendo personas con discapacidad.

Diseñar para la diversidad y heterogeneidad garantiza una experiencia más inclusiva.

### ◆ 5.2 Velocidad de conexión

- La velocidad de carga influye directamente en la usabilidad.
- Un sitio lento genera frustración y abandono.
- Ejemplo: **Netflix** permite descargar contenido y usar la app sin conexión, optimizando la accesibilidad.

### ◆ 5.3 Uso de estándares externos (W3C)

Los **estándares web** (World Wide Web Consortium, W3C) definen reglas sobre cómo crear y estructurar documentos en la red.

**Ventajas:**

- Mejor rendimiento y compatibilidad.
- Código más limpio y mantenible.

- Posicionamiento mejorado (SEO).

## Herramientas:

- [W3C Validator](#) — comprueba errores de código.
- [Pingdom Tools](#) — mide velocidad y rendimiento.

No siempre se pueden cumplir todos los estándares; depende del público objetivo y del presupuesto del desarrollo.

## 5 Navegación web

### ◆ Principios básicos

- Menús **claros y concisos** (máx. 7 elementos).
- Incluir enlace a **Inicio / Home** o botón de “Volver”.
- Incorporar buscador o sistema de ayuda.
- Enlaces coherentes y fácilmente identificables.

### ◆ Miga de pan (Breadcrumb)

Elemento de navegación que indica la **ruta jerárquica** dentro del sitio.

Facilita al usuario saber dónde se encuentra y volver atrás fácilmente.

Muy útil para usuarios con baja alfabetización digital.

### ◆ Rutas amigables

URLs cortas, comprensibles y descriptivas.

#### Ejemplo:

- ✓ [www.tienda.com/zapatos/mujer](http://www.tienda.com/zapatos/mujer)  
✗ [www.tienda.com/index.php?id=4532](http://www.tienda.com/index.php?id=4532)

#### Buenas prácticas:

- Describir el contenido.
- Usar guiones en URLs compuestas.
- Evitar caracteres extraños.
- Mantener consistencia en todo el sitio.

## 6 Cookies

Archivos que almacenan información sobre el usuario (navegador, idioma, procedencia, etc.).

Sirven para ofrecer un servicio personalizado, pero afectan la privacidad.

### ◆ Regulación

Desde **mayo de 2018**, el RGPD obliga a informar y solicitar consentimiento del usuario.

Rechazar cookies puede limitar la funcionalidad o personalización de la web, lo que impacta en la usabilidad.

## 7 Pautas de accesibilidad al contenido web (WCAG)

Las **WCAG (Web Content Accessibility Guidelines)** del W3C explican cómo hacer accesible el contenido digital.

Benefician tanto a personas con discapacidad como a usuarios con limitaciones temporales (brazo roto, conexión lenta, edad avanzada).

### ◆ Dirigidas a:

- Desarrolladores de contenido web.
- Diseñadores de sitios web.
- Desarrolladores de herramientas de evaluación y autor.

## 8 Pautas WCAG (resumen ampliado)

Nº	Pauta	Explicación breve
1	<b>Proporcione alternativas equivalentes para contenido visual o auditivo</b>	Ej. subtítulos, descripciones o texto alternativo.
2	<b>No se base solo en el color</b>	Permitir comprensión sin depender de colores.
3	<b>Utilice marcadores y hojas de estilo</b>	Separar contenido y presentación (HTML + CSS).
4	<b>Identifique el idioma usado</b>	Facilita a lectores de pantalla el cambio automático.
5	<b>Cree tablas que se transformen correctamente</b>	No abusar de tablas con fines estructurales.
6	<b>Verifique compatibilidad con nuevas tecnologías</b>	Mantener compatibilidad con navegadores antiguos.
7	<b>Controle el contenido dependiente del tiempo</b>	Evitar textos o animaciones en movimiento sin pausa.
8	<b>Accesibilidad de interfaces incrustadas</b>	Los objetos embebidos deben ser accesibles o tener alternativa.
9	<b>Diseñe para independencia del dispositivo</b>	Compatible con teclado, ratón, voz o dispositivos de asistencia.
10	<b>Use soluciones provisionales</b>	Versiones alternativas accesibles si no se cumple W3C.
11	<b>Use tecnologías y pautas W3C</b>	Cumplir especificaciones oficiales.
12	<b>Proporcione contexto y orientación</b>	Facilitar la comprensión a usuarios con dificultades cognitivas.
13	<b>Proporcione mecanismos claros de navegación</b>	Barras, mapas del sitio, buscadores.
14	<b>Documentos claros y simples</b>	Lenguaje comprensible y diseño limpio.

El nivel de accesibilidad depende del presupuesto del proyecto. Si no se pueden cumplir todas las pautas, debe alcanzarse el máximo grado posible dentro de las posibilidades técnicas.

## 9 Consorcio World Wide Web (W3C)

Organización internacional fundada por **Tim Berners-Lee**, inventor de la web.

Su misión es desarrollar **protocolos y estándares** que garanticen el crecimiento y compatibilidad del entorno web.

Fuente: <https://www.w3c.es/Consorcio>

## 10 Herramientas y test de verificación

- **Accesibilidad:** [W3C Validator](#)
- **Velocidad:** [Pingdom Tools](#)

Ambas permiten detectar errores y cuellos de botella en la carga o validación del sitio.

## 1 1 Por qué es importante

La accesibilidad **no solo beneficia a personas con discapacidad**, también a:

- Personas mayores.
- Usuarios con limitaciones temporales.
- Quienes tienen conexión lenta o dispositivos antiguos.

La **flexibilidad** en el diseño es clave para satisfacer diversas necesidades y contextos de uso.

## 1 2 Conclusión y conexión con la siguiente clase

En esta unidad hemos visto cómo la **usabilidad y accesibilidad** son pilares para el diseño centrado en el usuario.

Aplicar las **pautas WCAG** y los **principios de Nielsen** garantiza que nuestras interfaces sean inclusivas, intuitivas y coherentes.

→ En la siguiente clase aprenderemos a **diseñar un Wireframe**, aplicando los principios vistos hoy para construir interfaces accesibles y eficientes desde la planificación inicial.